

Appendix

実録『画像処理システムができるまで』

――プロジェクト成功の秘けつは「滞らない，手戻りがない」作業分担

編集部

本特集のために画像処理システムを設計するはめになった第３章と第４章の筆者二人．はっきりとした仕様が定まっていないのに，納期は２ヵ月後と決まっている．そして２ヵ月後，そのシステムは一発完動．ここでは，実際に二人がどのように作業分担し，設計を進めていったか，そのやりとりの一部を紹介する．そこには，手戻りなく，効率良く設計を行うためのヒントが隠されているかもしれない．なお，本稿は開発時の筆者らの打ち合わせやメールによるやりとりを編集部で再構成し，後日筆者らが注釈を加えたものである．（編集部）

● 事の発端

本誌編集者：「機能がわかりやすいシステムLSIを題材に，実例を交えたフレッシャーズ向けの特集記事を企画していますが，協力していただけませんか？」

森岡氏 ：「機能がわかりやすい…．例えば，なんらかの画像処理を行う回路をFPGAに実装して，ディスプレイに表示させるというのはどうでしょう．システムLSIということであれば，画像処理IPコアのほかに，インターフェース回路とかメモリ・コントローラなんかも実装して」

本誌編集者：「たしか，利用可能なPCIやSDRAMコントローラのIPコア[編集部注1]があったはずです」

：「それを利用しましょう」

鈴木氏 ：「私が画像処理のコア部分を作りましょう．森岡さんは，システム全体をインテグレートするということでどうでしょう」

：「わかりました．では，そうしましょう」

本誌編集者：「ありがとうございます．ちなみに，原稿の締め切り（4ヵ月後）を考えると，開発期間は2ヵ月程度になると思いますが…」

 ：「…」

 **のホンネ**：ひと事のように「システムLSIを作りましょう」と言ってみたものの，普通は大人数で半年～1年かけて作るのだから「二人で2ヵ月で完成」という要求はちょっとひどいなぁ…．やる事を相当絞り込まないといけないぞ．とりあえずシステムLSIではなくてシステムということにしよう[森岡氏注1]．言い出しっぺなので実機の実装役になったけど，回路が動かなかったとき，自分の担当部分が原因となる可能性がもっとも高くなるということか…．

 **のホンネ**：時間がないということで，つい自分にわかりやすい範囲を分担としてしまった．ごちゃごちゃとした部分は森岡氏任せで悪かったかなあ…．

● コア間のインターフェース仕様を決める

非常に大ざっぱなミーティングは終了し，システムの仕様（機能）の決定が両氏に委ねられることになった．

：「画像処理IPコアとほかのコアをどのようにつなげるかを考えてみました[森岡氏注2]．きちんとした決定はFPGA

編集部注1：PCIターゲット回路とSDRAMコントローラについては，来栖川電工の設計データを改造のうえ利用した．同社は，これらの設計データの使用範囲をCQ出版社の刊行物のみに定めており，商業目的の利用や再配布は認めていないので注意していただきたい．

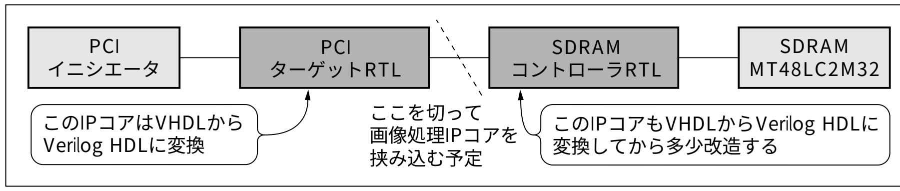
森岡氏注1：CPU＋メモリ＋IPコアが1チップになったのが，いわゆるシステムLSI．なので正確には，今回作成したものはシステムではあるがシステムLSIではない．実際にシステムLSIを作ろうとすると，搭載CPUやソフトウェア開発環境も調達しなければならないので，たいへんになる．

森岡氏注2：この時点で，PCIとSDRAMのインターフェースについてシミュレーション環境を構築して動作確認している．第3章に書いたとおり，今回いちばんネックになりそうなのがPCIやSDRAMといったインターフェース部である．ここでは，それらをきちんと作れそうか，まずあたりをつけようとしている．

KeyWord：IPコア，PCIターゲット，SDRAMコントローラ，空間フィルタ，レベル変換，階調ヒストグラム，MATLAB，FPGA，非同期リセット，同期リセット

図1 インターフェース仕様決め

PCIターゲット回路とSDRAMコントローラは，既存のIPコア（VHDL）をVerilog HDLに変換したうえで多少改造して利用する．さらに，「PCI経由でSDRAMにアクセスする」というデモ回路（既存のもの）を改造し，PCI側とSDRAM側に切り離して，今回設計する画像処理IPコアを挟み込む．



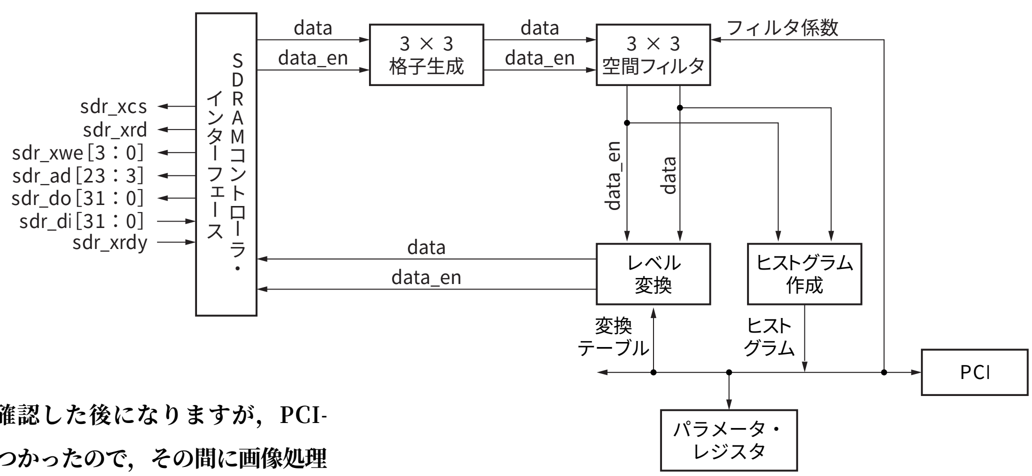
（**a**）インターフェースのイメージ

```
input   [25:0]  PCI_A;      // PCIアドレス（うちメモリ・アドレスはビット3～23，2Mワード×32ビット）
input   [31:0]  PCI_D_I;    // 書き込みデータ入力
output  [31:0]  PCI_D_O;    // 読み出しデータ出力
input           nPCI_CS;    // SDRAMアクセス・リクエスト
input           nPCI_RD;    // PCIリード・イネーブル
input           nPCI_WE0;   // PCI_D［07：00］ライト・イネーブル
input           nPCI_WE1;   // PCI_D［15：08］ライト・イネーブル
input           nPCI_WE2;   // PCI_D［23：16］ライト・イネーブル
input           nPCI_WE3;   // PCI_D［31：24］ライト・イネーブル
output          nPCI_RDY;   // SDRAMアクセス・アクノリッジ
```

（**b**）SDRAMコントローラのentity

図2 画像処理IPコアの仕様（一部）

画像処理IPコアのデータフローを示したもの．



評価ボードを使って動作を確認した後になりますが，PCI-SDRAMの接続デモ回路が見つかったので，その間に画像処理IPコアを挟み込む形になると思います（**図1**）．それから，今回はバースト転送はなしにします」


：「バースト転送なしということは，完全に1ワード単位でのアクセスということでよろしいですね？」


：「今のところそのつもりです森岡氏注3」


：「（**図1**（**b**）のSDRAMコントローラの）1アクセス・サイクルの完了はnPCI_RDYの立ち上がりをトリガにするということでいいんですね？」


：「はい，そうです森岡氏注4．画像処理IPコアからSDRAMコントローラを操作する場合，

1）CSを“L”に下げることで，バス・サイクル・スタート
2）RDY＝“L”になるまで待ち，“L”になったらSDRAMアクセス完了
3）リード時はデータも得られる
4）画像処理IPコアがCSを“H”に戻す
5）RDY＝“H”に戻るまで待つ．戻ったらサイクル終了

となるかと思います．普通のハンドシェイクです」

それから数日後．


：「画像処理IPコアの仕様（**図2**）とレジスタ設定一覧を準備しておきました森岡氏注5．画像処理IPコアで行う処理内容は，3×3格子による空間フィルタとテーブルを用いたレベル変換，階調ヒストグラムの3種類の機能です．空間フィル

森岡氏注3：この後，実機でデータ転送をテストしたところ，あまりにも遅く，実際にはバースト転送を使うように修正せざるをえなくなった．
森岡氏注4：いきなり細かい話に入っているが，ほんのちょっとした勘違いで結合時にまったく動かなくなるので，双方慎重になっている．
森岡氏注5：これは，システムの全体構成がどのようなものであるか，両者が暗黙のうちに共有しているということ．話が非常に速く進んでいる．

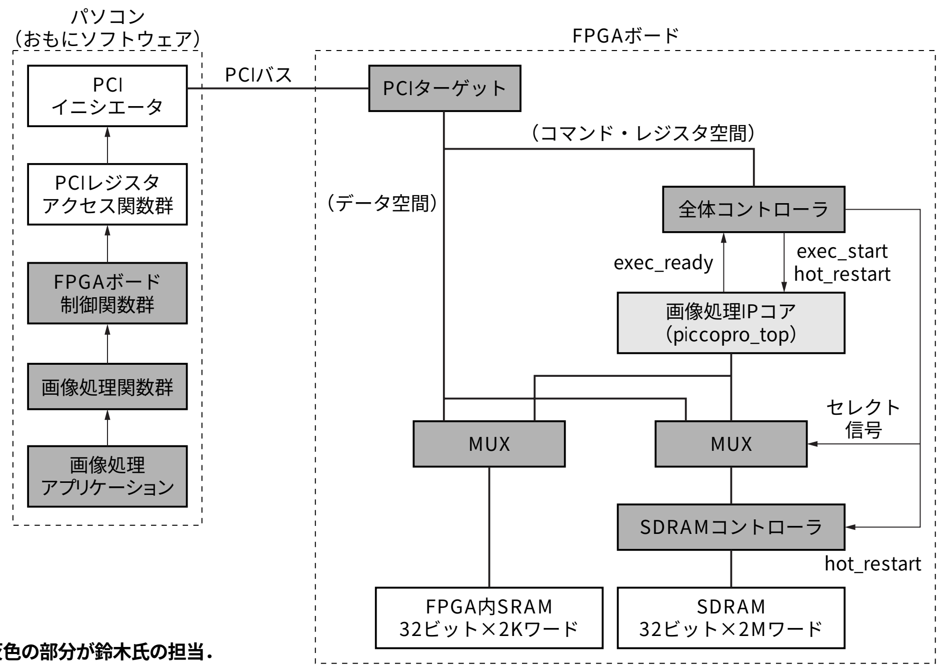


図3
システムの全体構成（変更前）
濃い灰色の部分が森岡氏の担当，薄い灰色の部分が鈴木氏の担当．

森岡氏はSRAM本体を自分が持つつもりだったので，このような入出力方向になっている．しかし実際には，SRAMは鈴木氏が持つことになったので，このin/outが逆になった

```
module piccopro_top(略);
input           clk;          //コア動作クロック．SDRAMと同じ66MHz
input           rst_x;        //パワーONリセット，負論理，非同期
input           hot_restart;  //ホット・リスタート，正論理，1パルス長，同期
input           exec_start;   //処理スタート，正論理，1パルス長
output          exec_ready;   //処理レディ，正論理（動作中のときに“L”）

output  [20:0]  sdram_a;      //SDRAMアドレス（32ビット・ワード単位）
input   [31:0]  sdram_d_rd;   //SDRAM読み出しデータ
output  [31:0]  sdram_d_wr;   //SDRAM書き込みデータ
output          sdram_rd_x;   //SDRAMリード・イネーブル，負論理
output  [3:0]   sdram_we_x;   //SDRAMライト・イネーブル，負論理
output          sdram_req_x;  //SDRAMアクセス・リクエスト，負論理
input           sdram_ack_x;  //SDRAMアクセス・アクノリッジ，負論理

output  [10:0]  param_a;      //SRAMアドレス（32ビット・ワード単位）
input   [31:0]  param_d_rd;   //SRAM読み出しデータ
output  [31:0]  param_d_wr;   //SRAM書き込みデータ
output          param_wr_x;   //SRAMリード/ライト・モード（リードが“H”）
```

図4
森岡氏提案のインターフェース仕様の一部
（画像処理IPコアの処理entity）

タの係数設定やテーブルの設定などによって処理のバリエーションが変わります」



：「IPコアに多少手を入れたものの，こちらも実機でパソコンからPCIを経由してSDRAMの読み書きが行われることを確認できたので森岡氏注6，システムの全体構成をまとめました（**図3**）．ドキュメントのとおりのentityにしていただければつながります（**図4**）」



：「（ドキュメントを見て）私はFPGA内蔵のSRAM（32ビット×2K）は画像処理IPコアの中に持つものと考えていました森岡氏注7．ヒストグラム用のメモリと階調変換用テーブルはFPGA内蔵SRAMに割り当てますが，これらはPCIと比べものにならないくらい頻繁にアクセスします．なので，フロアプランニングなどを考えても手元においておきたいんです．また，フィルタ係数などはメモリに割り当てていないので，これもSRAMとして外に持つのは厳しいところです．ものによっては並列に参照する必要がありますので…．（**図4**の）param_a,

森岡氏注6：じつは森岡は，PCIとSDRAMの実機動作が確認できた時点で，少なくとも自分の作成部については「まずまちがいなくいける」と確信していた．

森岡氏注7：複数の回路モジュールが共有して使うメモリやレジスタについては，その本体をだれ（どの回路モジュール）が持っておくかが問題になる．これをへたに決めると，切り替えインターフェースが複雑になって，いざ回路モジュールを統合したときにバグだらけになる．ここではまさに，そのクリティカルな話に突入したところ．

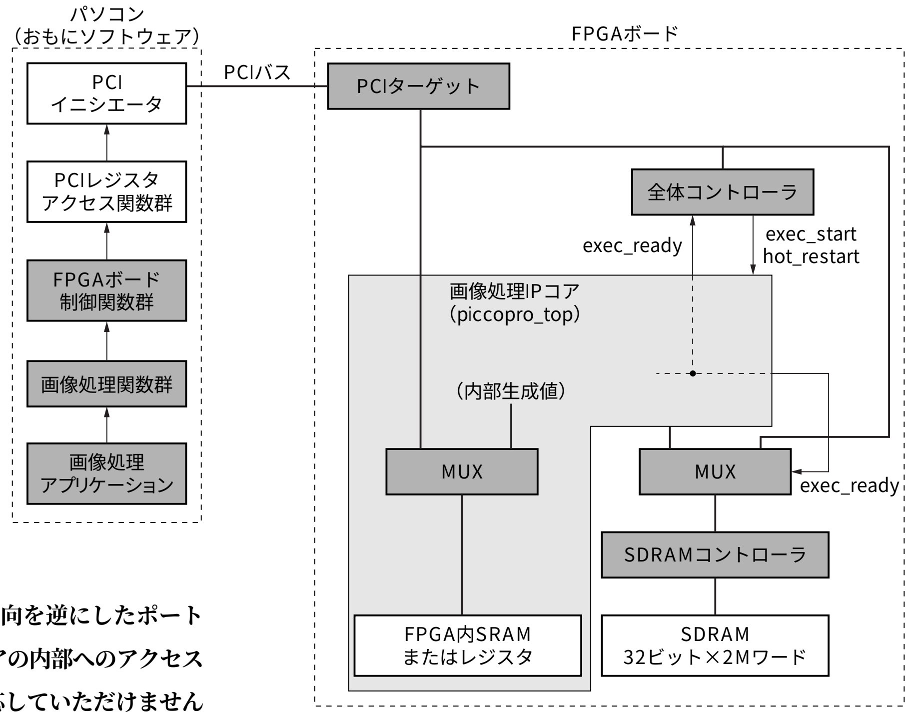


図5
システムの全体構成(変更後)
濃い灰色の部分が森岡氏の担当，薄い灰色の部分が鈴木氏の担当．FPGAのSRAMは画像処理コア内に持つことにした．

param_d_rd，param_d_wrの入出力の方向を逆にしたポートを用意していただければ，画像処理IPコアの内部へのアクセスはできるようにしておくので，それで対応していただけませんか？ SRAMはこちらで対応します」

:「では，そうしましょう．ただ，SRAMを鈴木さんの画像処理IPコア内部へ入れるなら，SDRAMも内部へ入れていただくほうが，制御信号の混乱が生じないのでベターだと思います森岡氏注8．両方内部にあれば，セレクタ切り替えの制御信号は画像処理IPコアから外へ出す必要がなくなります．SRAM制御を鈴木さん，SDRAM制御を私というように分担すると，動作上，かなりクリティカルな部分なのでくいちがいが起こるかもしれませんし，シミュレーションでもつらくなりそうです」

:「うーん，ここは一つ譲ってもらえないでしょうか？ SRAMは画像処理IPコア内部，SDRAMは外部というところで…．PCIターゲットからMUX経由でSRAMへ行っているラインをじかに画像処理IPコアに接続するだけなので，むしろ経路としてはシンプルになるように思えるのですが…．いかがでしょうか？」

:「では，おっしゃるとおりにいたします(**図5**)．セレクタの切り替えはexec_readyで行うことにすれば，変更は最小でいけると思います」

:「ありがとうございます．強引ですみません…」

のホンネ：この時点でもっとも恐れていたのは，二人の回路を統合したときに動かなくなること．それゆえ，なるべく切り口をシンプルにしようといろいろ考えていた．また，鈴木氏と共同で回路を作るのは初めてだったため，互いの進めかたがマッチするかを図りながら発言していた．実際には，本業のしごとでもめったにないくらいパッパッと話が通じている．

のホンネ：回路構成の簡素化や検証の容易性を考慮して提案した構造だったが，異なる意見になったことに気づく．バックグラウンドの知識の違いも感じる．互いに正論を語っても意見はまとまらない．このあたりの回路設計は手慣れているのでまず問題は起こさないだろうし，検証が複雑にならないためにも，「ここは一つ譲ってください！」．

● **メモリ・タイミングの取りかたについてのやり取り**

:「すみません，1点忘れていました．ヒストグラムを作成した際のパソコンからのSRAM森岡氏注9の読み出しですが，アドレスを確定してからデータが確定するまでに最低

森岡氏注8：と言ってはみたものの，後でじっくり考え直してみたところ，SDRAMの端子はチップ外に出ていくので，どのみち自分が管理しなければならず，自分の最初の案が有利とは言えないことに気づいた．

森岡氏注9：これは画像処理IPコア内部に持つパラメータ設定用のSRAMのこと．パソコンからこのSRAMへ各種パラメータを設定する．そのとき，PCIバスからメモリ・アクセス信号がやって来るが，SRAMの読み書きをしている間，パソコンがきちんと待ってくれるのかを話している．

2クロックかかります．PCI側からそのような細かい制御は可能ですか？」

：「処理が終わったことがパソコンへ伝わって，パソコンがメモリ読み出しを始めるまでに2クロック以上，それどころかμs～msオーダはかかります．なので，現実問題としては放っておいても問題ないと思います」

：「すみません，言いたいことがうまく伝わっていませんでした．でき上がったヒストグラムによってパソコンが階調変換テーブルを作成するため，パソコンがFPGA上のSRAMの内容を読み出す必要があります．全部で256階調×3色＝768アドレス分あります．この読み出しのときの話です．たぶん，1アドレスずつPCI経由で読み出すことになると思うのですが，FPGAのSRAMポートに読み出しアドレスが伝わってデータが出るまでに2クロック以上かかるという意味です」

：「あ，意味がわかりました．SRAMのレイテンシ分，PCIは待たせます．これは私のほうで管理するつもりです．ただし，SRAMですがFPGAでは1クロック・レイテンシにも2クロック・レイテンシにもできますが，どちらにします？」

：「2クロックで設計しています．このレイテンシ＋1クロック（メモリ出力のMUX）で3クロックのレイテンシを見込んでおいてください」

の追補：ここでは細かい話をしているが，回路モジュール接続部のタイミングをきちんと打ち合わせないと，回路を統合したときにまったく動かないので，二人とも慎重になっている．

の心中：ヒストグラムの読み出しにおけるPCIからのSRAMアクセスはアドレス量が多い．少しでも性能を稼ごうとして高速なレスポンスを期待されてもFPGAの特性上不可能な側面もあるので，念のため確認を行った．森岡氏はきっとわかっているだろうとは思っていたが，メモリのレイテンシまで気づかってもらい，確実に意思の疎通が図れている．ああ，私は幸せ者だ．

● 検証の進めかたについてのやり取り

：「画像処理IPコアがつながるようにインターフェース部やメモリ・コントローラのRTLを改変したうえで，仮の画像処理IPコアでシミュレーション，実機とも動作確認しました．これで，鈴木さんから画像処理IPコアのソース・コード（RTL）を待つのみという状況になりました森岡氏注10．今回は共同設計ということで，鈴木さんから設計物（完成品）をまとめていただくのではなく，なるべく密に情報交換しながら進めるほうがよいと思っています．なお，66MHz＋α（80MHz～100MHz程度）のタイミング制約に合わせた状態でコードをいただけると助かります」

：「タイミングに関する検証は行っておきます．ただし，制約を満たさない場合でも配置の悪さに起因する配線遅延によって占められているようなケースはこの段階では無視します．これは配置制約によって解決すべき問題なので，画像処理IPコア単品で解決しても意味がないと考えます鈴木氏注1」

：「私の担当分のロジックはターゲットとなるFPGA全体の4％しかありません森岡氏注11．したがって，全体を統合したときの配置配線は画像処理IPコア単体の場合と大差なく，コア単体であらかじめタイミング収束を済ませておいてもらえると，統合時もほぼ問題なく動作する公算が高いと思います森岡氏注12．私では画像処理IPコア内部に踏み込んだ最適化が行えないので，もし配置配線の問題でタイミングが収束しなかったとしても，鈴木さんから制約ファイルなどをもらわないと手も足も出ないと思います」

：「これはいつごろ必要でしょうか。10日ほど待っていただければ機能検証まで済ませたものをお送りできますが…」

：「部分的でもよいので早め早めに繰り返しスナップ・ショットをいただくほうが，問題の発見や手戻りを最小にできそうです森岡氏注13．『つながらない』というような致命的な問題が後からわかると悲惨ですし…」

森岡氏注10：この時点で，自分とほかの人が作ったモジュールをつなぎ合わせて動くかどうかが，最大かつ最後の難関として残っている．

鈴木氏注1：FPGAにおいて，一度タイミング制約を満たした設計が，わずかな変更を加えただけで制約を満たさなくなることは珍しくない．原因は多くの場合，配置配線が大幅に変わってしまうためで，これは配置配線アルゴリズムのよしあしによる．したがって，森岡氏の回路が小規模のものであっても，これを加えることでタイミング検証の結果は変化するであろうと考えてこのように提案した．この段階では，論理合成のみによるタイミング推定で配置配線の影響を排除しておき，RTLによって解消できるタイミングの不ぐあいを検証しておきたい（配置配線によるタイミングの不ぐあいは，基本的にRTLでは解消できない）．下位モジュールごとに相対配置制約を定義する方法もあるが，集積度が上がってくると互いに成立しないという問題も起きてくる．

森岡氏注11：これほど余裕があるのは非常に珍しいケース．実際の設計では，残りエリアに余裕がないケースのほうがずっと多い．エリアが足りなくなってくると，配置配線後にクロック周波数が上がらなくなるなど，二重苦三重苦の状況に追い込まれやすい．なので，このときは「鈴木氏もそれが身に染みていて，慎重な物言いになっているのだろう」と思いながら森岡は発言していた．

森岡氏注12：これはあくまでも，残りエリアが96％もあるのでできる発言．

森岡氏注13：経験的に，状況が厳しいときほどエンジニアどうしのコミュニケーションは密に行い，こまめに調整を図るとよい．

リスト1　非同期リセット

```
always@( posedge clk or negedge rst_x) begin
    if (~rst_x)              //初期化処理1
        else begin
            if (hot_restart) //初期化処理2
            else
            …中略…
end
```



：「ではPCIターゲットからの読み出し/書き込みを確認するための（演算はやらないがインターフェース部だけがある）ダミーの回路モジュールを本日中に作成します森岡氏注14．今回の構成では，（IPコア内部で使う）おおよそのモジュールがそろわないことには意味のある検証を行えないので，（森岡氏のモジュールとの接続チェックには）ダミーのIPコアを新しく作るしかないと思います．しかし，SDRAMへの出し入れの確認は，やはりIPコア全体の設計が終わるまで待ってください森岡氏注15．これについては，ダミーを作るよりも現物のIPコアを作るほうが効率が良いと思います」



の追補：自分以外の人が作った回路と自分の回路をつなぎ合わせるのは，何度経験してもドキドキする瞬間．今回のように，早め早めに確認をとっていくことと，最初から細部をきっちり詰めておくことが一発完動のコツだろう．逆に，最初に「なあなあ」でやって，最後に現物で合わせ込もうとすると，失敗しやすい．



の追補：グループ・ワークにおいて，自分の担当分をまちがいなくこなすことはたいせつだが，ほかのメンバの手を止めないように配慮することもプロジェクトの進行上必要とされる．個人レベルの少しの手間がプロジェクト全体に大きな影響を与えることは珍しくない．

【検証こぼれ話】

検証についてのやり取りの最中，森岡氏の「MATLABを使ってチェックしてみます」という発言から話はちょっと横道に…．



：MATLABが使える環境があるのはうらやましいですね．



：最近は，この手のツールを使わないとテスト一つ簡単にできなくなってきていて，なかなかたいへんです．笑い話のようですが，回路にはバグがないけれどテスト環境がバグだらけで，どちらをデバッグしているのかわからないようなことも実際のしごとでありました．



：それ，よくあるんですよね．検証の計画が甘いときに，外付けモジュールのビヘイビア・モデルを慌ててでっち上げたりすると，どっぷりとはまってしまったりして…．

リスト2　非同期リセットで誤作動が起こる可能性がある場合

```
reg  [1:0]  register;
always @(posedge clk or negedge rst)
    if (!rst)
        register <= 2'b00;
    else
        register <= 2'b11;
    end
```

● リセットの考えかたで盛り上がる



：「いただいたダミーの回路モジュールはうまくつながってPCIバスから叩けました．ですから，本番のIPコアもまず問題なくつながると思います．ところで細かいことですが，PCIやSDRAMのコアが非同期リセットなので，**リスト1**のような非同期リセットに修正をお願いできますか森岡氏注16．本音を言うと私は『同期でいきたい派』なのですが…」



：「私もです．ただ，もしトラブルのもとになるとすると，積極的に直していくしかないかもしれませんね．リセットのパスをクリティカル・パスで扱うことになるんでしょうか」



：「FPGAやCPLDだと，非同期preset/clrは普通global lineになりますよね？ なので，放っておいても同時性（全フリップフロップで同時にリセットがデアサートされること）はわりと確保できるかもしれません．ただ，クロックとの位相の関係で（クロックの直前・直後での）リセットのリリース時はやはり心配です．私の経験では，ASICなどでは非同期リセット信号はクロック・ツリーと同じで特別扱いですね」

森岡氏注14：複数人（あるいは複数チーム）がかかわる実際のシステムLSI設計では，各モジュールの内部設計に入る前に，このような仮モジュール（スタブと言う）を全体の管理者へ提出して，相互接続の問題がないかを事前に確認することが多い．ハードウェアだけでなく，ソフトウェア開発の場合も同様．この少しの手間で，モジュール作成後にインターフェース部の設計ミスが発覚し，全部が作り直しになるといったリスクをかなり防止できる．

森岡氏注15：実際にはこの後で，森岡が持っていたSDRAMコントローラのRTLデータとそのテスト環境を鈴木氏に渡し，鈴木氏のIPコアでSDRAM読み書きができることを，まず鈴木氏のほうで検証してもらうことになった．このように，複数人での開発作業ではなるべく同じテスト環境を使うようにしておくと，モジュール統合時のトラブルが少なくなる．

森岡氏注16：この時点では，森岡側が非同期リセット，鈴木氏側が同期リセットになっていた．リセットを同期で行うべきか非同期で行うべきか，あるいは折衷案でいくのかは，大昔から行われている宗教論争の一つ．とりあえず，同一チップ内部ではなるべく同じ方式で統一するのが望ましい．

:「非同期リセットの場合，二つのDFF（Dフリップフロップ）でリセット・デアサートのタイミングがずれて，その間にクロック・エッジが入ってしまうと，最初の有効なクロック・エッジがそのDFF間で1クロック分ずれることになります．これでトラブルを起こさないためには，非同期リセットを使う回路はリセットのデアサート・タイミングに対して鈍い必要がありますね」

:「同感です．**リスト2**のような回路だと，デアサート直後に“01”や“10”になる可能性があってまずいです．多くの回路では，デアサート直後は動かずにアイドル・ステートに入るので，こうした問題は顕在化しにくいのかもしれません」

:「こういう話はあまり議論されませんね」

:「私もきちんと議論したいほうなのですが，わりと『デザイン・ルールだから』のひと言で片づけられてしまうことが多い…」

:「おお！ 伝家の宝刀『デザイン・ルール』．確かにこれには勝てませんね」

● **つなげたら動いちゃいました**

毎日のように今回のシステム設計について話を詰めていった二人だが，編集部からの設計依頼から2ヵ月後（正確には1ヵ月と10日後！），めでたくすべての動作について実機確認を終えた．

:「鈴木さんからもらったテスト・シーケンスをプログラム化して，すべて実機でpassしました！ 境界値テストを行っていないので完全とは言い切れませんが，おおむね問題なく動いていると思います．また，実画像の処理例もいくつか取れました．実画像として512画素×512画素の画像サイズのものを使い，処理速度は約2.4Mピクセル/sでした」

:「これは計算どおりでしたね．3×3フィルタが1画素当たり28クロックなので，66M÷28＝2.35Mとなります．それにしてもこの短期間で実機動作をクリアしてしまうとは，ひとえに森岡さんの的確な采配と深夜に及ぶ（？）努力のおかげですね」

:「実際のしごとでこうしたプロジェクトを始めると，けっこう長時間にわたってすったもんだしてしまいがちですが，今回はまれに見るスムーズさでした．阿吽（あうん）の呼吸で話がぽっぽっと通じていくので，非常に楽に作業が進みました」

:「不思議とうまくいきましたね．あははは！」

:「はい．全体の統合でもう少しスリリングな展開があるかなと覚悟していましたが，くっつけるやいなや動いてしまいました(^^;」

の感想：なにはともあれ，ほぼ一発完動だったので安心した．実機が動く瞬間というのは，何回経験してもうれしいものだ．本稿のように複数の筆者が共同で設計するケースは珍しいと思う．

の感想：森岡さんにはめんどうな部分をまとめてもらっただけでなく，こちらのつごうにも気を使ってもらって感謝．実機が動く瞬間に立ち会えなかったのが残念．

本誌編集者のホンネ：かなりむちゃなお願いをしてしまい，もし途中でお二人から「やっぱり（2ヵ月でシステム設計は）ムリ！」と言われたらどうしようと内心オロオロしていた．が，あれよあれよという間に仕様が固まり，実機動作確認に至った．経験を積んだエンジニアというのはスゴイ！ と感嘆した．と同時に，「これで特集が落ちなくてすむ」と胸をなで下ろした．